

15-104 Introduction to Computing for Creative Practice

Fall 2021

16 Objects

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1



Variables

- In Javascript, we have seen that we can store a single data value in a variable.


```
var x = 15104;
var winner = true;
var boxColor = "purple";
```
- We can also reference an array from a variable.


```
var temps = [79, 81, 57, 64, 63, 57, 58]
```
- But we also saw multiple variables holding data for the same object we were drawing on the canvas (e.g. drawing one box on the canvas):


```
var x, y, dx, dy, c;
```
- Although they're individual variables, they're all connected to the same shape.

2

Objects

- In Javascript, an **object** can be defined using a variable. However, the object has properties and behaviors associated with the object.

- An object can be defined literally

```
var sqr = {x: 100, y: 100, w: 50, dx: 5,
          r: 255, g: 255, b:0};
```

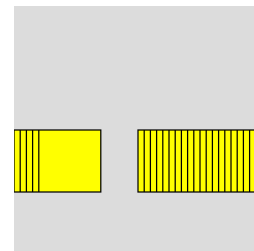
- To access any properties of the object, we use dot notation, listing the object variable name followed by a dot followed by the property (field) of the object.

```
fill(sqr.r, sqr.g, sqr.b);
rect(sqr.x, sqr.y, sqr.w, sqr.w);
sqr.x += sqr.dx;
if (sqr.x > width) { sqr.x = -sqr.w };
```

3

A square object

```
var sqr = {x: 100, y: 100, w: 50, dx: 5, r: 255, g: 255, b:0};
function setup() {
  createCanvas(200, 200);
  background(220);
  frameRate(30);
}
function draw() {
  fill(sqr.r, sqr.g, sqr.b);
  rect(sqr.x, sqr.y, sqr.w, sqr.w);
  sqr.x += sqr.dx;
  if (sqr.x > width) { sqr.x = -sqr.w; }
}
```



4



Objects

- To access any properties of the object, you can also treat the object variable like an array and use the field name, in quotes, as the index:

```
fill(sqr["r"], sqr["g"], sqr["b"]);
rect(sqr["x"], sqr["y"], sqr["w"], sqr["w"]);
sqr["x"] += sqr["dx"];
if (sqr["x"] > width) { sqr["x"] = -sqr["w"]; }
```

- We will prefer the dot notation since this is common across languages.

5



Objects

- We can also create objects by construction within our program code.

```
var sqr;
function setup() {
  createCanvas(200, 200);
  background(220);
  frameRate(5);
  sqr = new Object();
  sqr.x = 100; sqr.y = 100; sqr.w = 50; sqr.dx = 5;
  sqr.r = 255; sqr.g = 255; sqr.b = 0;
}
```

6

Objects and Equality

- Consider the following Javascript code:

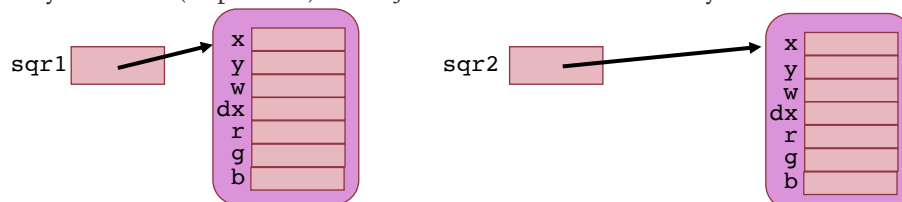
```
var sqr1;
var sqr2;
function setup() {
  sqr1 = {x: 100, y: 100, w: 50, dx: 5,
         r: 255, g: 255, b:0};
  sqr2 = new Object();
  sqr2.x = 100; sqr2.y = 100; sqr2.w = 50; sqr2.dx = 5;
  sqr2.r = 255; sqr2.g = 255; sqr2.b = 0;
  print(sqr1 == sqr2);
}
```

- Although both objects are exactly the same, this prints `false`. Why?

7

Objects and Equality

- Each variable points to (“references”) its own object.
- This reference is typically a memory address where the object is stored.
- The operator `==` is testing the contents of the variables `sqr1` and `sqr2` which are memory addresses.
 - Remember, these object variables do not hold the objects themselves. They reference (or point to) the objects somewhere in memory.



8

Objects and Equality

- Consider the following Javascript code:

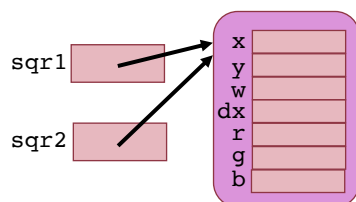
```
var sqr1;
var sqr2;
function setup() {
  sqr1 = {x: 100, y: 100, w: 50, dx: 5,
         r: 255, g: 255, b:0};
  sqr2 = sqr1;
  print(sqr1 == sqr2);
}
```

- Now, this code prints `true`. Why?

9

Objects and Equality

- Using the reference principle, when we assign `sqr2` with the value stored in `sqr1`, `sqr1` contains the location of the object (not the object itself).
- So `sqr2` gets a copy of the location of the object.
- Now `sqr1` and `sqr2` are both referencing (pointing to) the same object.
- This reference is typically a memory address where the object is stored.



10

Aliasing

- In the previous example, `sqr2` is an alias of `sqr1`.
 - They both essentially point to the same thing.
- If we change the object through one alias, we will see that change through the other alias.

```
sqr1 = {x: 100, y: 100, w: 50, dx: 5,
        r: 255, g: 255, b:0};
sqr2 = sqr1;
sqr2.w = 75;
print(sqr1.w);      // prints 75
```

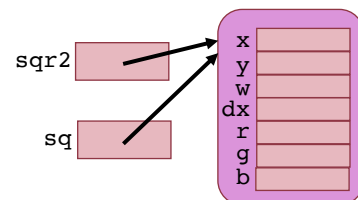
11

Objects and Function Calls

- Whenever you call a function with a parameter, a copy of the argument is passed to the function and stored in the parameter.
- The same happens with objects, so when you call a function and pass an object, you're really passing a reference to it which is stored in the parameter.
- So the parameter acts like an alias of the original object variable.

```
function draw_sqr(sq) {
    fill(sq.r, sq.g, sq.b); ...
}
```

```
draw_sqr(sqr2);
```



12

Two square objects

```

var sqr1 = {x: 100, y: 100, w: 50, dx: 5, r: 255, g: 255, b:0};
var sqr2 = {x: 50, y: 50, w: 50, dx: 10, r: 0, g: 255, b:255};


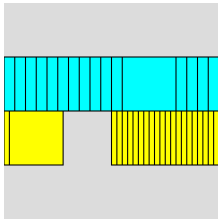
function setup() {
  createCanvas(200, 200);
  background(220);
  frameRate(5);
}

function draw() {
  draw_sqr(sqr1);
  draw_sqr(sqr2);
}

function draw_sqr(sq) {
  fill(sq.r, sq.g, sq.b);
  rect(sq.x, sq.y, sq.w, sq.w);
  sq.x += sq.dx;
  if (sq.x > width) { sq.x = -sq.w; }
}

```

*sq is an alias,
first for sqr1,
then for sqr2*

13

Arrays of Objects

- You can even have an array of objects!


```

var sqr_array = [];


function setup() {
  sqr_array[0] = {x: 100, y: 100, w: 50, dx: 5,
                 r: 255, g: 255, b:0};
  sqr_array[1] = {x: 50, y: 50, w: 50, dx: 10,
                 r: 0, g: 255, b:255};
  ...
}

function draw() {
  fill(sqr_array[0].r, sqr_array[0].g, sqr_array[0].b);
  square(sqr_array[0].x, sqr_array[0].y, sqr_array[0].w);
  ...
}

```



14




Many Boxes (again)

```

var bx = []; // array of boxes
function setup() {
  createCanvas(200, 200);
  for (i = 0; i < 100; i++) {
    bx[i] = new Object();
    bx[i].x = random(width);
    bx[i].y = random(height);
    bx[i].dx = random(-5, 5);
    bx[i].dy = random(-5, 5);
    bx[i].c = color(random(255), random(255), random(255));
  }
  frameRate(5);
}

```

15



Many Boxes (again)

```


function draw_box(b) {
  fill(b.c);
  rect(b.x, b.y, 10, 10);
}

function update_box(b) {
  b.x += b.dx;
  b.y += b.dy;
  if (b.x > width) b.x = 0;
  else if (b.x < 0) b.x = width;
  if (b.y > height) b.y = 0;
  else if (b.y < 0) b.y = height;
}

```

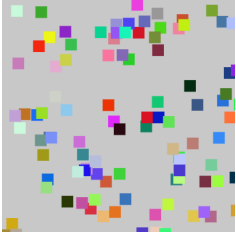
These functions run some generic box object **b**. (**b** is an alias.)
 Note there is no subscript since this should work for any of the box objects.

16




Many Boxes (again)

```
function draw() {
  background(200, 200, 200);
  noStroke();
  for (i = 0; i < 100; i++) {
    draw_box(bx[i]);
    update_box(bx[i]);
  }
}
```



For each of the 100 box objects, send the i^{th} box object to the two helper functions to draw and update each box.

17



Try This

- Modify the horizontally moving box program so that the box reappears at some new random vertical position when it reappears on the left side.
- Modify the many boxes programs so that all of the boxes get new random colors when the mouse is clicked.

18